

**APPLICATION
FOR
UNITED STATES LETTERS PATENT**

CERTIFICATE OF MAILING

TITLE: INTERRUPT REQUEST CONTROLLER
APPLICANT: KENDELL A. CHILTON

"EXPRESS MAIL" Mailing Label Number E1245468142US

Date of Deposit September 29, 1999
I hereby certify under 37 CFR 1.10 that this correspondence is being
deposited with the United States Postal Service as "Express Mail
Post Office To Addressee" with sufficient postage on the date
indicated above and is addressed to the Assistant Commissioner for
Patents, Washington, D.C. 20231.

Kevin Donnelly
Kevin Donnelly

INTERRUPT REQUEST CONTROLLER

Background of the Invention

5 This invention relates generally to data storage systems, and more particularly to data storage systems having redundancy arrangements to protect against total system failure in the event of a failure in a component or subassembly of the storage system.

10 As is known in the art, large mainframe computer systems require large capacity data storage systems. These large main frame computer systems generally include data processors which perform many operations on data introduced to the computer system through peripherals including the data storage system. The results of these operations are output to peripherals, including the storage system.

15 One type of data storage system is a magnetic disk storage system. Here a bank of disk drives and the main frame computer system are coupled together through an interface. The interface includes CPU, or "front end", controllers (or directors) and "back end" disk controllers (or directors). The interface operates the controllers (or directors) in such a way that they are transparent to the computer. That is, data is stored in, and retrieved from, 20 the bank of disk drives in such a way that the mainframe computer system merely thinks it is operating with one mainframe memory. One such system is described in U.S. Patent 5,206,939, entitled "System and Method for Disk Mapping and Data Retrieval", inventors Moshe Yanai, Natan 25 Vishlitzky, Bruno Alterescu and Daniel Castel, issued April 30, 1993, and assigned to the same assignee as the present invention.

As described in such U.S. Patent, the interface may also include, in addition to the CPU controllers (or

PENDING PAPERLESS

directors) and disk controllers (or directors), addressable cache memories. The cache memory is a semiconductor memory and is provided to rapidly store data from the main frame computer system before storage in the disk drives, and, on 5 the other hand, store data from the disk drives prior to being sent to the main frame computer. The cache memory being a semiconductor memory, as distinguished from a magnetic memory as in the case of the disk drives, is much faster than the disk drives in reading and writing data.

10 The CPU controllers, disk controllers and cache memory are interconnected through a backplane printed circuit board. More particularly, disk controllers are mounted on disk controller printed circuit boards. CPU controllers are mounted on CPU controller printed circuit 15 boards. And, cache memories are mounted on cache memory printed circuit boards. The disk controller, CPU controller and cache memory printed circuit boards plug into the backplane printed circuit board. In order to provide data integrity in case of a failure in a controller, the 20 backplane printed circuit board has a pair of buses. One set the disk controllers is connected to one bus and another set of the disk controllers is connected to the other bus. Likewise, one set the CPU controllers is connected to one bus and another set of the CPU controllers is connected to 25 the other bus. The cache memories are connected to both buses. Each one of the buses provides data, address and control information.

Summary of the Invention

. In accordance with the present invention, an 30 interrupt request controller is provided for processing a plurality of interrupt logic signals. The controller includes: a programmable bit masking section fed by the interrupt logic signals, adapted to mask selected ones of

SEARCHED
INDEXED
SERIALIZED
FILED

the interrupt signals; a interrupt priority section fed by the programmable mask section for coupling unmasked ones of the interrupt signals to a plurality of outputs selectively in accordance with a predetermined priority criteria.

5 In one embodiment, the request controller processing a plurality of interrupt logic signals and includes: a programmable section fed by the interrupt signals, for selecting assertion sense and/or assertion type of each one of the interrupt signals.

10 In accordance with one feature of the invention, an interrupt request controller is provided for processing a plurality of interrupt logic signals. The controller includes: a programmable section fed the interrupt signals, for storing a bit for each one of the interrupt logic
15 signals representative of whether the logic state of the interrupt logic signal should be, or should not be, inverted and for producing a corresponding output logic interrupt signal in accordance therewith.

20 In accordance with another feature of the invention, an interrupt request controller is provided for processing a plurality of interrupt logic signals. The controller includes: a programmable section fed the interrupt signals, for storing a bit for each one of the interrupt logic signals representative of whether the logic state of the
25 interrupt logic signal should remain as an edge or be converted to a level and for producing a corresponding output logic interrupt signal in accordance therewith.

In accordance with still another feature of the invention, an interrupt request controller is provided for
30 processing a plurality of interrupt logic signals. The controller includes: a programmable section fed by the interrupt signals, for selecting assertion sense and/or assertion type of each one of the interrupt signals; a

CONFIDENTIAL - SECURITY INFORMATION

programmable bit masking section coupled to the programmable assertion sense/assertion type section, adapted to mask selected ones of the interrupt signals; a interrupt priority section fed by the programmable mask section for coupling 5 unmasked ones of the interrupt signals to a plurality of outputs selectively in accordance with a predetermined priority criteria.

In one embodiment, the programmable assertion sense and/or assertion type section includes for each one of the 10 interrupt logic signals an interrupt sense register for storing a bit representative of whether the logic state of the interrupt logic signal should be, or should not be, inverted.

In one embodiment, the programmable assertion sense 15 and/or assertion type section includes for each one of the interrupt logic signals, an interrupt type register for storing a bit representative of whether the logic state of the interrupt logic signal should remain as an edge or be converted to a level.

20 Brief Description of the Drawing

These and other features of the invention will become more readily apparent from the following detailed description when read together with the accompanying drawings, in which:

25 FIG. 1 is a block diagram of a data storage system according to the invention;

FIG. 2 is a block diagram of an exemplary one of a plurality of directors used in the system of FIG. 1, such director having a central processing unit in accordance with 30 the invention;

FIG. 3 is a block diagram of a microprocessor interface used in the central processing unit of the director of FIG. 2;

FIGS. 3A through 3C are block diagrams of an XCVR core and a CPU XCVR used in the microprocessor interface of FIG. 3, FIG. 3A showing the data rebuffering mechanism which supports microprocessor read operations, FIG. 3B showing the 5 data rebuffering mechanism that supports microprocessor read operations, and FIG. 3C showing data rebuffering mechanism that supports the microprocessor address path;

FIG. 3D is a diagram useful in understanding an addressing feature provided by the microprocessor interface 10 of FIG. 2 in accordance with the invention;

FIG. 4 is a block diagram of a main memory interface according to the invention used in the microprocessor interface of FIG. 3;

FIG. 5 is a block diagram of an error detector and 15 corrector according to the invention used in the main memory interface of FIG. 4;

FIG. 5A is a block diagram of an XOR arrangement used in the error detector and corrector of FIG. 5;

FIG. 6 is a block diagram of an interrupt request 20 controller used in the microprocessor interface of FIG. 3;

FIG. 7 is a block diagram of an interrupt inverter register used in the interrupt controller of FIG. 6;

FIG. 8 is a block diagram of an interrupt type 25 register used in the interrupt request controller of FIG. 6; and

FIG. 9 is a diagram of a fault detector adapted to detect hard faults on a bi-directional data line according to the invention.

Description of the Preferred Embodiments

30 DATA STORAGE SYSTEM

Referring now to FIG. 1, a data storage system 10 is shown wherein a host computer 12 is coupled to a bank 14 of disk drives through a system interface 16. The system

interface 16 includes a cache memory 18. A plurality of directors 20₀-20₁₅ is provided for controlling data transfer between the host computer 12 and the bank 14 of disk drives as such data passes through the cache memory 18. A pair of 5 high address busses TH, BH is electrically connected to the high address memory section 18H of cache memory 18 as described in U. S. Patent application Serial No. 09/223,115 entitled "Data Storage System", inventors D. Castel et al, filed December 30, 1998, assigned to the same assignee as 10 the present invention, the entire subject matter thereof being incorporated into this application by reference. A pair of low address busses TL, BL is electrically connected to the low address memory section 18L of cache memory 18. The cache memory 18 has a plurality of storage location 15 addresses. Here, the storage locations having the higher addresses are in the high address memory sections 18H and the storage locations having the lower addresses are in the low address memory sections 18L. It should be noted that each one of the directors 20₀-20₁₅ is electrically connected 20 to one of the pair of high address busses TH, BH and one of the pair of low address busses TL, BL. Thus, each one of the directors 20₀-20₁₅ is able to address all locations in the entire cache memory 18 (i.e., to both the high address memory sections 18H and the low address memory sections 18L) 25 and is therefore able to store data in and retrieve data from any storage location in the entire cache memory 18.

More particularly, a rear-end portion of the directors, here directors 20₀-20₁₁, is electrically connected to the bank 14 of disk drives and a front-end portion of the 30 directors, here directors 20₈-20₁₅, is electrically connected to the host computer 12.

In operation, when the host computer 12 wishes to store data, the host computer 12 issues a write request to

PCTD200000000000

one of the front-end directors 20₈-20₁₅ to perform a write command. One of the front-end directors 20₈-20₁₅ replies to the request and asks the host computer 12 for the data.

After the request has passed to the requesting one of the 5 front-end directors 20₈-20₁₅, the director determines the size of the data and reserves space in the cache memory 18 to store the request. The front-end director then produces control signals on either a high address memory bus (TH or BH) or a low address memory bus (TL, BL) connected to such 10 front-end director depending on the location in the cache memory 18 allocated to store the data and enables the transfer to the cache memory 18. The host computer 12 then transfers the data to the front-end director. The front-end director then advises the host computer 12 that the transfer 15 is complete. The front-end director looks up in a Table, not shown, stored in the cache memory 18 to determine which one of the rear-end directors 20₀-20₇ is to handle this request. The Table maps the host computer 12 address into an address in the bank 14 of disk drives. The front-end 20 director then puts a notification in a "mail box" (not shown and stored in the cache memory 18) for the rear-end director which is to handle the request, the amount of the data and the disk address for the data. Other rear-end directors poll the cache memory 18 when they are idle to check their 25 "mail boxes". If the polled "mail box" indicates a transfer is to be made, the rear-end director processes the request, addresses the disk drive in the bank, reads the data from the cache memory and writes it into the addresses of a disk drive in the bank 14. When data is to be read from the disk 30 drive to the host computer 12 the system operates in a reciprocal manner.

DIRECTOR

Referring now to FIG. 2, an exemplary one of the directors, here director 20₀, is shown. The director 20₀ has an X CPU section 22 and a Y CPU section 24 which share shared resources 40 such as flash memories, etc. The flash memory stores the BIOS, or boot-up routine, for the CPU sections 22, 24. The X and Y CPU sections 22, 24 are identical in construction, the X CPU section 22 being shown in more detail in FIG. 2. Here, the X CPU section 22 shown 10 is a rear end director and thus is coupled to the disk drives 14 (FIG. 1), it being understood that had such section been in a front end director such X CPU section 22 would have been connected to the host computer 12. The X CPU section 22 is also coupled to the cache memory 18 (FIG. 15 1), as indicated.

Referring in more detail to the X CPU section 22, it is noted that such section 22 includes a Direct Memory Access (DMA) section 42 which is an interface between the cache memory 18 (FIG. 1), the bank of disk drives 14 and the central processing unit 44 of the X CPU section 22. The central processing unit 44 includes a microprocessor 46, here a Power PC microprocessor, a main memory 48, a CPU decoder 50 (e.g., a programmable logic device), and a microprocessor interface 52, here an Application Specific Integrated Circuit (ASIC). The microprocessor interface 52 will be described in more detail in connection with FIG. 3. Suffice it to say here, however, that the microprocessor interface 52 is a comprehensive Power PC microprocessor support integrated circuit chip having several discrete functional sections, including a main memory interface 54 having a set of registers 53, a data rebuffering section 56, an interrupt request (IRQ) controller 58, and an embedded Static Random Access Memory (SRAM) 60. Here, the main

memory 48 is an SDRAM, however, as will be described, other types of memories may be used such as a RAMBUS DRAM (RDRAM).

Here, the main memory interface 54 is adapted to manage one or two banks of main memory 48 SDRAMs, providing up to 128 MB of memory I/O space using 64-megabit RAM densities. The data is Error Correction Code (ECC)-protected and single-bit errors can be corrected as they are detected. The main memory SDRAM interface 54 fully supports byte, half-word and word reads and writes through built in Read-Modify-Write cycles.

The interrupt request (IRQ) controller 58 provides flexible interrupt management. Here, the interrupt request controller 58 supports up to 28 external interrupts and 4 fatal interrupts as well as internal interrupts. These interrupts can be assigned to any interrupt level by programming level and mask registers which will be described in more detail in connection with FIG. 6. In addition, the interrupt request controller provides facilities to assist in Motorola 68060-style interrupt generation as will also be described in more detail in connection with FIG. 6.

The data rebuffering section 56 of the microprocessor interface 52 provides dual-bus address and data rebuffering. Fully flexible and controlled by the external CPU decoder 50, the address and data paths are rebuffed to enable connection to local and shared resources 58. The options of registering data, assembly/disassembly function, and parity generation are controlled by the logic in the external CPU decoder 50.

The microprocessor interface 52 also provides facilities to capture address and data ranges, and to provide interrupts upon capture. This capability is useful for debugging operations to trap rouge address or data cycles.

CENTRAL PROCESSING UNIT 44

Referring now also to FIGS. 2 and 3, the central processing unit 44 (FIG. 2) of the director 20, includes, as noted above, a microprocessor 46; a main memory 48; a CPU decoder 50; and a microprocessor interface 52, here an ASIC. The data rebuffering section 56 of the microprocessor interface 52 is adapted to couple data from one of a plurality of data ports, here data port A, data port B, and data from the embedded SRAM 60 and interrupt request controller 58, to a data port of the microprocessor 46 selectively in accordance with a control signal supplied by the CPU decoder 50. The main memory interface 54 is adapted for providing control signals to the main memory 48 section for enabling data transfer between the main memory 48 and the microprocessor 50 through the data rebuffering section 56.

As noted above, the main memory 48 is a selected one of a plurality of memory types, such as an SDRAM or an RDRAM. Each memory type has a different data transfer protocol. The main memory interface 54 is configured in accordance with the selected one of the plurality of memory types to provide a proper memory protocol to data being transferred between the microprocessor 46 and the main memory 48 through the main memory interface 54. As noted above, one main memory type is an SDRAM and another main memory type is a RDRAM, it being understood that other types may also be used.

Referring now also to FIG. 3, the main memory interface 54 is shown in more detail to include a main memory API controller 64 and a microprocessor memory interface control/EDAC section 66. The data to and from the main memory 48 passes through the EDAC portion 70 of section 66, such EDAC portion being described in more detail in

- 20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
- connection with FIG. 5. The main memory API controller 64 is responsible for translating upper-level read, write, and refresh requests into the appropriate low-level control signals for the type of main memory 48 being used. That is,
- 5 the data to and from the main memory 48 is through the main memory API controller 64 in response to control signals supplied by the microprocessor memory interface control/EDAC section 66. The control signals are generic, that is they are independent of the type of main memory 48 being used.
- 10 The main memory API control 64 is hard-wired a priori to translate the generic control signals into the proper protocol for the particular type of main memory 48, i.e., SDRAM or RDRAM, etc. If for example, the original application for the CPU controller 44 is with an SDRAM, and
- 15 at some future time the application will use an RDRAM, the only portion of the microprocessor interface 52 which must be reconfigured is the main memory API controller 64; the design of the microprocessor memory interface control/EDAC section 66 may remain unchanged.
- 20 Referring also to FIG. 4, the microprocessor memory interface 66 has three sections; the EDAC section 70, a memory interface control section 72 and an address/data register and control section 74. The microprocessor memory interface 66 manages the interface between the
- 25 microprocessor 46 and the main memory API controller 64. It performs the required address decoding, burst address counter, Opcode generation, refresh generation, and main memory API controller 64/microprocessor 46 synchronization.
- The address/data register and control section 74
- 30 contains the necessary input and output registers to properly align the microprocessor 46 data stream with the main memory 48 data stream after it has passed through the EDAC section 70.

The EDAC section 70 is described in more detail below in connection with FIG. 5.

Various control and data lines and busses are shown in FIG. 4:

5 CPU_Dat1Op<71.0>
DBBnp
CPU_Adrp<35.0>
TSnp
ABBnp
10 TT1p
TT3p
TBSTnp
TSIZp<2.0>
AACKnp
15 TAnp
which are defined in the Motorola MPC 750 microprocessor Manual;
Clock
Reset and
20 Pulse
which are system inputs;
the following EDAC signals:
SBE - single bit error indication
MBE - multiple bit error indication
25 PErr - parity error indication
Syndrome - XOR reduced syndrome bits
Err_Cnt - error count (number of EDAC errors detected during a data transfer)
Config_Data - indicates where the EDAC is either an
30 error detect or error correct mode and whether the EDAC is in even or odd parity;
the following are defined by Rambus Application Guide (www.rambus.com):

WD
RD
Wdone
Rrdy
5 Busy
Last
Ai
Ao
Mo

10 Op
Start

and the following which are conventional SDRAM
interface signals:

15 DQ<71.0>
Adr<13.0>
CSn
RASn
CASn
WEn
20 DQM
CKE

Referring again to FIG. 3, the data rebuffering
section 56 is responsive to a control signal from the CPU
decoder 50, for coupling data between a selected one of the
25 data ports, i.e., port A, port B, or the embedded SRAM or
the interrupt request controller 58, and the data port of
the microprocessor 48. More particularly, the data
rebuffering section 56 has a port A transceiver (XCVR), a
port B XCVR 82, an XCVR core 83, and a CPU XCVR 84, arranged
30 as indicated in FIG. 3. The XCVR core 83 is a selector
which, in response to a control signal from the CPU decoder
50, couples data/address between port 86 of the CPU XCVR 84
and either: the port A XCVR 80; or, the port B XCVR 82; the

embedded SRAM 60, or the interrupt request controller 58 selectively in accordance with a control signal fed to the XCVR core 83 from the CPU decoder 50. (As noted from FIG. 2, here the port A XCVR is connected at port A to the cache 5 memory 18 (FIG. 1) through the DMA 42 and the port B XCVR 82 is coupled at port B to the shared resources 40.

The CPU XCVR is a data distribution unit having a plurality of ports each one of the ports being coupled to a corresponding one of: (i) the XCVR (selector) 83; (ii) the 10 Synchronous DRAM 60; (iii) the interrupt request controller 58; (iv) the microprocessor 46 data port; and (v) the main memory interface 54.

More particularly, the XCVR core 83 and CPU XCVR 84 are shown in FIGS. 3A, 3B, and 3C. The data rebuffering 15 section 56 mechanism that supports microprocessor write operations is shown in FIG. 3A. The here 72 bit data from the microprocessor 46 data transfer bus transfers to the microprocessor interface 52 at the CPU_DataIOP interface. The microprocessor interface 52 has two registers 87, 89, 20 one for the upper data word and one for the lower data word, respectively. The CPU_DatWUClkEnp and DatWLClkEnp are used to enable data registering into the upper and lower word lanes on the rising edge of a clock, not shown, respectively. Parity is stored along with the corresponding 25 word's data. CPU_DatSyncSelp, when clear, causes the input registers 87, 89 to be by-passed providing an asynchronous data path to Port A and Port B. CPU_ULSelp determines whether the upper or lower register 87 or 89 word is passed to port A or port B. This is necessary as the 30 microprocessor 46, here Power PC, data interface 52 is 72 bits while here the Port A and Port B interfaces are 36 bits wide.

Port A and Port B are identical in operation. Thus
PortA/PortB_DatWClkEnp clocks data
PortA_DatIOp/PortB_DataIOp into the output port data
registers 187, 189 respectively on the rising edge of the
5 Clock. PortA/PortB_DatSyncSelp, when clear, provides an
output register bypass mechanism for asynchronous output
data. PortA/PortB_DatIOOEnp enable the output drivers 91,
93 for the ports A and B.

A separate data path (SDIO) is used for the main
10 memory interface 54. The main memory interface 54 has a
dedicated set of input registers 95, 97 in the CPU XCVR 84
which are enabled (EN) when DBBnp from the CPU decoder 50 is
asserted.

The data rebuffering section 56 mechanism that
15 supports the microprocessor 46 read operation is shown in
FIG. 3B. For read operations, data transfers to the
microprocessor interface 52 at the PortA/PortB_DatIOp
interfaces. PortA/PortB_DatRClkEnp are used to enable data
registering on the rising edge of the Clock, respectively.
20 PortA/PortB_DatSyncSelp, when clear, causes the input
registers 99, 101 to be bypassed providing an asynchronous
data path. Such an arrangement allows design flexibility
(e.g., tradeoff between performance and latency).

CPU_DatPortSelp determines the data source for the
25 CPU output registers 103, 105. The Table below list the
possible data sources:

CPU_DatPortSelp Value	Data Source
000	Data currently stored in the microprocessor output registers
001	Data currently stores in the Port A output registers
5 010	Data Currently stored in the Port B output registers
011	All 1s with good parity
100	Internal configuration and status registers. The register data is presented on the lower word and the upper word is padded with all 0s and good parity.
101	Data from Port A.
110	Data from Port B.
10 111	Data from the EDAC 70

CPU_DatRUClkEnp and CPU-DatRLClkEnp clocks data into the upper and lower word lanes (e.g., registers 103, 105) on the rising edge of the Clock. The external CPU decoder 50 15 (FIG. 2) must control these signals appropriately to present the PortA/PortB data on the correct word line based on the least-significant address decodes. CPU_DatSyncSelp, when clear, causes the CPU output registers 103, 105 in the data path to be bypassed providing asynchronous output to the 20 Power PC microprocessor 46.

A separate data path SDIO is used for the main memory interface. The output from the main memory controller 64 is fed to a top-level output multiplexer 107. Any time the microprocessor interface 52 decodes an address 5 in its internal main memory space, the data presented to the microprocessor is from the main memory controller. Otherwise, the data presented to the microprocessor is from the microprocessor 46 output structure.

CPU_DatIOOEnp controls the CPU output buffers and is 10 valid only during non-main memory transactions. During main memory control, the microprocessor interface 52 controls these output drivers 103 to present data to the microprocessor 46 at the correct time.

Referring now to FIG. 3C, the addressing rebuffering 15 mechanism that supports the microprocessor address path is shown. Address rebuffering in the microprocessor interface 52 is unidirectional and is source at the CPU_Adrp port. CPU_AdrSyncSelp determines on a global basis whether the address path is synchronous-pipelined or asynchronous 20 (default). During asynchronous operation, the external CPU decoder 50 (FIG. 2) must ensure that the address has enough time to propagate through the microprocessor interface 52 to Port A/PortB before asserting Ext_AACKnp.

During synchronous operations, TSnp clocks the 25 current address into the CPU Address input register 111. When the external CPU decoder 50 (FIG. 2) asserts Ext_AACKnp, the address previously registered into the CPU Address input register 111 is transferred to the Port A and Port B Address Output registers 113, 115. 30 PortA/PortB_AdrIOOEnp selectively enable their respective port address output drivers 117, 119.

The internal main memory controller 64 has a dedicated set of address registers 111. When TSnp asserts,

the address on CPU_Adrp is clocked into one of the main memory address registers 111 for use by the main memory controller.

As noted briefly above, the main memory 48 has a plurality of data storage sections, one section having a first set of addresses and a second section having a second set of addresses. It should be understood that there may be more than one second section (i.e., the second section may be made up of a plurality of regions, or sections). The microprocessor interface 52 produces addresses for the main memory 48. The main memory controller 64 has a decoder 65 responsive to the produced addressed to determine whether the produced address is within the first set or the second set of addresses. The main memory interface 54 is adapted for coupling to the main memory 48. The main memory 48 is adapted for coupling to the microprocessor 46 and to the data rebuffering section 56. The main memory interface 54 provides control signals for enabling data transfer between the main memory 48 and the microprocessor 46 through the data rebuffering section 56. The main memory interface 54 is responsive to the decoder 50 and enables the second section in the memory 48 when the decoder 50 determines the produced address is in the second set of addresses and the first section is enabled for addressing by the produced address when the main memory interface 54 determines the produced address is in the first set of addresses.

More particularly, and referring to FIG. 3D, four external SDRAM chip selects CS0-CS3 provide access to the SDRAM 48. Access to the SDRAM via the chip select is decoded by the external CPU decoder 50 (FIG. 2). The decoded address must fall outside the memory space allocated to the microprocessor interface 52, here addresses 000 0000h to 7FF FFFFh. A configuration register defines an address

PENDING
DECODED
00000000

mask which is applied to the Power PC microprocessor 46 address to qualify a segment size, i.e., the size of the SDRAM associated with each chip select. Each chip select also has a segment address register, (i.e., DRAM CSx segment 5 address) with which it is associated. This segment address register provides an OR-field which is added to the microprocessor address after the address segment size mask is applied. This allows the four chip selected SDRAM spaces to be located independently throughout the SDRAM space.

10 Assuming, for example, a full memory configuration of 128 MB and that this configuration demonstrates four segments of 64K apiece located at the top 256 K of SDRAM memory space. Assume that the CPU decoder decodes a chip select memory access at address 0820 1234h. As will be 15 described in connection with FIG. 3D, the input address provided by the microprocessor 46, here 0820 1234h and stored in register 111 will be mapped to a new output address, here 07FC 1234h) shown in region 212 shown in FIG. 3D.

20 More particularly, the microprocessor address segment mask of 07FF 0000h is applied to this address yielding a resulting address of 0800 1234h. Next, the appropriate DRAM chip select segment address is OR'd in. Assuming chip select CS0 was used, 07FC 0000h is OR'd with 25 the interim result to produce 0FFC 123h as the chip select memory address. The uppermost 5 bits are truncated from the address as they are beyond the 128 MB of addressable memory. The final memory address becomes 07FC 1234h for the chip select of the SDRAM access.

30 More particularly, and referring to FIG. 3D, it is first noted that first addresses in the main memory 48 (i.e., the section 216 having the first 128 MBytes minus 256k of memory) is dedicated for addressing only by the

PENTIUM® PROCESSOR

microprocessor interface 52. The upper 256 Kbytes have been segmented into four segments 214a-214d of 64k each. Each one of these four segments 214a-214d is selected by a corresponding one of four chip selects CS0-CS3, as indicated. Here, the address (here, for example, 0820 1234h) is shown provided by the microprocessor 48 is stored in register 111 (FIG. 3C). The segment size is stored in one of the registers, here register 200) in set of registers 53 (FIG. 2) in the main memory interface 54, here, in this example, 07FF 000h). For each bit in register 111 is gated with a corresponding bit in register 200 in gates 202 with the gated output (i.e., interim result) being shown in region 206, here 0800 1234h. The desired chip select segment enabling the various storage sections for the main memory 48 is stored in register 208 in set of registers 53. Here, in this example the register 208 stores 07FC 0000h, as indicated. Each one of these bits is OR gated with a corresponding one of the bits produced at the output of gates 202 to provide the result shown in region 210, here the chip select SDRAM mapped addresses OFFC 1234h, as indicated. After truncating bits 31 through 37, as indicated by the bracket 212, such truncated result is fed as the address to the main memory 48, as indicated. It is noted that addresses 07FC 0000 through 07FF FFFF are in the above described second section of the main memory 48 here having four region 214a-214d, each with 64K addresses. Here, regions 214a-214d are selected by chip selects CS0-CS3, respectively. Thus, it is noted that the input address provided by the microprocessor 46 and stored in register 111 has been mapped to a new output address shown in region 212.

ERROR DETECTOR

Referring now to FIG. 5, the error detection and correction section (EDAC) 70 is shown in more detail. The

EDAC 70 is coupled between the main memory API (Application Programming Interface) controller 64 and the CPU XCVR 84 through a data bi-directional I/O bus (MDIO) and the system data bi-directional I/O bus (SDIO), as shown in FIG. 3.

5 Considering first data flow from the SDIO to the MDIO, a source of data, here the data read from the CPU XCVR 84 (FIG. 3), is applied to the SDIO. The data has a plurality of bytes, here eight bytes (i.e., 64 bits), each one of the bytes having a parity bit. A parity checker 90 is

10 responsive to the parity bits on SDIO and the eight bytes of data on SDIO for detecting a parity error in the data eight bytes of data produced by the data source on SDIO. Any parity error is reported to the CPU decoder 50 (FIG. 2).

A pair of check bit generators are provided. Each

15 one of the parity bit generators 92, 94 is fed the plurality of bytes, here eight bytes, of the data on SDIO. Each one of the parity bit generators 92, 94 is a redundancy code generator, here a modified Hamming code check bit generator, which generates a plurality of check bits from the plurality of bytes fed thereto. A logic, here an XOR gate 96, is fed by the check bits produced by the pair of check bit generators 92, 94 for determining whether the pair of check bit generators 92, 94 produce different check bits for the same bytes fed thereto. A logic, here an XOR gate

20 arrangement 98 is fed by the check bits produced by one of the pair of check bit generators, here generator 92, for producing a single, combined parity bit from the generated check bits. One such XOR gate arrangement 92 is shown in FIG. 5A to include a plurality of, here six XOR gates 93

25 arranged as shown being fed by eight bits B_1-B_8 to produce a combined parity bit on line 95.

The eight parity bits on SDIO are fed to a logic 100, here an XOR gate arrangement similar to the arrangement

98 shown in FIG. 5A, for producing a combined parity bit CHK_PAR from the plurality of parity bits. A logic 102, here an XOR gate, is provided to determine whether the combined parity bit produced by the logic 98 and the 5 combined parity bit produced by the logic 100 have the same logic state. Absent any error or fault, the parity bit CHK_PAR and the combined parity bit produced by the XOR arrangement 98 will be the same. The output of the XOR gate 96 and XOR gate 102 are fed to an OR gate 104 which produces 10 a check bit error signal for an OR gate 132 if a fault has been detected by either a mismatch in the parity bits fed to the XOR gate 96 or the XOR gate 102.

Considering now the flow of data from the MDIO to the SDIO, a data source, here the data read from the main memory 48 and placed on the MDIO bus, has a plurality of, here eight bytes and a plurality of, here eight check bits associated with such bytes. The eight check bits are stored with the data and had been generated from a redundancy generating code, here a modified Hamming code. A pair of 15 check bit generators 106, 108 are provided. Each one of the check bit generators 106, 108 is responsive to the plurality of, here eight bytes of the data on the MDIO bus for generating here eight check bits. The check bit generators 106, 108 are redundancy code generators, here modified 20 Hamming code generators. A logic 110, here an XORing arrangement such as that described in connection with FIG. 5A is provided for producing a combined parity bit from the generated check bits. The combined parity bit is fed to an 25 input of XOR gate 112.

30 A pair of error detector/correctors 114, 116 is provided. Each one of the detectors/correctors 114, 116 is responsive to: the eight bytes of data on the SDIO; the generated check bits of a corresponding one of the pair of

check bit generators 106, 108; and, the eight check bits associated with the bytes of data on the SDIO. Each one of the pair of error detector/correctors 114, 116 includes a pair of syndrome generator 118, 120, respectively, and a 5 pair of data_correctors 122, 124, respectively. Each one of the pair of error detector/correctors 114, 116 is adapted to detect an error in the bytes fed thereto and adapted to correct such data in event of a detected error in accordance with conventional modified Hamming code error correction and 10 detection techniques (i.e., a single bit error is detectable and correctable and a multiple bit error is detectable).

A parity bit generator 126 is provided for producing a plurality of, here eight, parity bits from data produced by one the pair of error detector/correctors 114, 116, here 15 the data_corrector 122 of generator 114. A logic 128, here an XOR arrangement similar to that described above in connection with FIG. 5A, is provided for producing a combined parity bit, GEN_PAR, representative of the eight parity bits produced by the parity bit generator 126. The 20 eight parity bits produced by the parity bit generator 126 are fed to the SDIO.

A logic, here the XOR gate 112, is provided to determine whether the combined parity bit produced by the logic 110 and the combined parity check bit from the logic 25 128 have the same logic state. A logic 130, here an XOR gate is provided for determining whether the data produced by the data_correctors 122, 124 of the pair or error detector/correctors 114, 116 is the same. If they are different an syndrome error is produced. This syndrome 30 error together with the output of the XOR 112 and the CB error are fed to an OR gate 132 to produce a chip error signal for the interrupt request controller 58 (FIG. 3).

INTERRUPT REQUEST CONTROLLER 58

Referring now to FIG. 6, the interrupt request controller 58 is shown in more detail. The interrupt request controller 58 manages interrupt inputs and programmable interrupt inputs or general purpose outputs.

5 Here 28 dedicated inputs feed the interrupt request controller 58 directly and four fatal interrupts (Fatal_IntIop <3..0>) are OR-reduced to a single entry by a fatal mask 300 in the interrupt request controller 58. It

10 is noted that these four fatal requests may be programmed as general purpose outputs.

The interrupt request controller 58 includes an interrupt inverter section 304 (FIG. 7) fed by the ORrd output from the fatal mask 300 and the here 31 other

15 interrupt request (IRQs); a watchdog IRQ produced by watchdog timer 301 which is fed refresh pulses, select signals and data from the XCORE selector 83, a Match IRQ produced by Match 303 which is fed addresses, data and select signals from the XCORE selector 83, a microprocessor

20 interface IRQ produced by the microprocessor interface 52 (FIG.2), and 28 Intp <27..0> IRQs produced by inputs external pins of the microprocessor interface 52. Each one of the 32 IRQs are fed to an interrupt invert register 306 in the interrupt invert section 304, and exemplary one

25 thereof being shown in FIG. 7. The interrupt request IRQ is fed to one input of an XOR gate 308, the other input being fed by the output of a register 310. If the particular IRQ sense (i.e., either positive-true or negative-true) is to be inverted, a logic 1 is stored in the register 310. On the

30 other hand, if the IRQ 306, is not to be inverted, a logic 0 is stored in the register 310.

In any event, the 32 outputs from the interrupt invert 204 are fed to an interrupt type register section

2000 DEPOSITED
1000 INDEXED
500 SERIALIZED
100 FILED

312. More particularly, each bit is fed to an interrupt type register 314, an exemplary one thereof being shown in FIG. 8. The interrupt type register 314 determines whether the interrupt IRQ should be an edge (leading or trailing edge) or remain as a level. More particularly, the register 314 includes a multiplexer 316. The multiplexer 316 has one port, port A, fed by the interrupt request IRQ and the other input, port B, fed by an edge logic 318. The edge logic 318 includes an OR gate 320. One input to the OR gate is fed by 5 the interrupt request IRQ through an AND gate 319 and register 321, as indicated, and the other input to the OR gate 320 is fed by a clear signal through AND gate 322, as indicated. The output of the OR gate is fed to a register 330. The output of register 330 is fed to the B input of 10 multiplexer 316. A register 326 is provided to produce the port select signal for the multiplexer 316. If a logic 1 is stored in the register 326, the A port is coupled to the output of the multiplexer 316 thereby selected a level as in 15 interrupt request type. On the other hand, if a logic 1 is stored in the register 326 the output of the edge logic 318 is coupled to the output of the multiplexer 316.

More particularly, in response to a clock pulse, register 321 stores the level of the interrupt fed thereto. The stored level is fed to the inverted input of AND gate 25 319. Thus, AND gate 319 compares the level of the present interrupt with the level of the previous interrupt (i.e., the level of the interrupt on the previous clock). If they differ, a logic 1 is asserted and becomes stored in register 330. The stored logic 1 in register 330 is then fed back to 30 the non-inverting input of AND gate 322. The output of AND gate 322, in the absence of a clear signal from the microprocessor interface 52 (FIG. 2) passes through OR gate 320 where it remains until the presence of a clear signal.

Thus the clear opens the feedback path. Thus, in response to an edge (i.e., leading or trailing edge) of the interrupt signal, such edge is converted into a level which appears at the output of register 330 (i.e., at the B input of selector 5 316.

The interrupt requests produced by the interrupt type register section 312 is fed to a current interrupt register 321. The contents of the current interrupt register may be observed for testing or, in operation, for 10 polling, for example. These contents are also fed to a source mask 323 wherein any one of the 32 bits may be masked as desired.

Additional registers assign each interrupt source to a particular level interrupt. A mask register allows 15 individual interrupts to be masked. In addition, the interrupt request controller contains logic for emulating the Motorola 68k family interrupt logic. This logic includes registers which contain current IPL level and status and provide the necessary hardware to simply 20 implement an emulated 68k interrupt scheme.

More particularly, the Int4 through Int0 mask registers in section 325 select which interrupt bits from the source mask register 323 participate on a given level. Each Intx mask is OR-reduced in OR gate section 327 to 25 arrive at the primary IRQnp<4..0> outputs. These outputs may be used where desired to change the flow in execution of a program, for example. Similarly, the Int68k7 through Int68k1 mask registers in section 329 select which interrupt bits participate on Motorola 68k-style level. Each level is 30 OR-reduced in section 331 and then fed to the IPL level/mask register 333. This register can individually mask any interrupt level. An IRQ68knp interrupt is then generated

based on the level information stored in the IPL level/mask register 333 and the current IPL register 335.

The microprocessor 46, here Power PC, microcode may use the IPL/mask and current IPL registers 333, 335 to emulate the built-in capabilities of the Motorola 68k family of microprocessors. The current register contains the highest-level interrupt currently pending. By reading this register 335 and then reprogramming the IPL level/mask register 333, the microprocessor interface 52 can emulate the IPL level masking capabilities found in the 68k family of microprocessors. A simple driver routine need only be written to manipulate these registers upon receiving an interrupt so the ISR can branch to the traditional 68k-style interrupt vectors.

Thus, section 333 is a programmable mask section for coupling unmasked ones of the interrupt signals to a plurality of outputs selectively in accordance with a predetermined priority criteria.

FAULT DETECTOR

Referring now to FIG. 9, a circuit 400 is shown for detecting a hard fault (i.e., a ground fault or a connection to a fixed, unchanging voltage) on a bi-directional data line 402. The circuit 400 includes an input/output terminal 304 connected to one end of the bi-directional line 402; a data receiver 406 having an input coupled to a second end of the bi-directional line 402 for receiving data on the terminal 404; an data driver 408 having an output coupled to second end of the bi-directional line 402 for producing data on the terminal 404; an XOR gate 410 having a pair of inputs, one being coupled to an output of the data driver 408 and the other being coupled to an input of the data receiver 406, for determining whether data produced by the data driver is received by the data receiver.

For example, assume a ground is shorted to the line 402. When the data driver 408 is enabled by an enable signal EN and is to produce, in response to the logic signal fed to the input of the data driver 408, a high logic
5 voltage on the line bus, the ground will produce a low logic signal at the output of the XOR gate 410. Thus, the XOR gate output will indicate that the input to the data driver 408 is different from the output produced by the data receiver 406, thereby producing a logic 1 and therefore indicating a
10 fault.

It should be understood that if the hard fault on the line 402 is a high voltage which is fixed and remains constant over a number of normal data cycles, when the data driver 408 is enabled by an enable signal EN and is to
15 produce, in response to the logic signal fed to the input of the data driver 408, a low logic voltage on the line bus, the high voltage fault will produce a high logic signal at the output of the XOR gate 410. Thus, the XOR gate output will indicate that the input to the data driver 408 is
20 different from the output produced by the data receiver 406, thereby producing a logic 1 and therefore indicating a fault.

Other embodiments are within the spirit and scope of the appended claims.

25 What is claimed is: